UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/583,097	08/02/1999	Marc Tremblay	004-1391-1	7166

22120 7590 08/15/2003

ZAGORIN O'BRIEN & GRAHAM LLP
401 W 15TH STREET
SUITE 870
AUSTIN, TX 78701

EXAMINER
HUISMAN, DAVID J

ART UNIT	PAPER NUMBER
2183	

DATE MAILED: 08/15/2003

14

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No. 09/583,097	Applicant(s) TREMBLAY, MARC
	Examiner David J. Huisman	Art Unit 2183

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *07 August 2003* .

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *9-36* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *9-36* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

11)☒ The proposed drawing correction filed on *07 August 2003* is: a)☐ approved b)☒ disapproved by the Examiner.

If approved, corrected drawings are required in reply to this Office action.

12)☐ The oath or declaration is objected to by the Examiner.

Priority under 35 U.S.C. §§ 119 and 120

13)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____ .

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

14)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application).

a) ☐ The translation of the foreign language provisional application has been received.

15)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449) Paper No(s) _____ .
4) ☐ Interview Summary (PTO-413) Paper No(s). _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: .

DETAILED ACTION

1. Claims 9-36 have been examined.

Papers Submitted

2. It is hereby acknowledged that the following papers have been received and placed of record in the file: #13. Amendment "C" with a 2-month extension as received on 8/7/2003.

Specification

3. The title of the invention is not descriptive. A new title is required that is clearly indicative of the invention to which the claims are directed.

4. The disclosure is objected to because of the following informalities: In the last sentence of the amended paragraph beginning on page 9, line 14, the second "more" is unnecessary.

Appropriate correction is required.

Drawings

5. The proposed drawing correction filed on August 7, 2003, has been disapproved because it is not in the form of a pen-and-ink sketch showing changes in red ink or with the changes otherwise highlighted. See MPEP § 608.02(v).

Maintained Rejections

6. The rejections set forth in the previous Office Action, paper #12, mailed on March 6, 2003, have not been overcome by Applicant and are hereby maintained by the examiner and included below for Applicant's convenience.

Maintained Claim Rejections - 35 USC § 102

7. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

8. Claims 9-36 are rejected under 35 U.S.C. 102(e) as being anticipated by Vegesna et al., U.S. Patent No. 5,488,729 (as disclosed by Applicant and herein referred to as Vegesna).

9. Referring to claim 9, Vegesna has taught a superscalar processor that performs, over plural execution cycles of the superscalar processor, instruction grouping for dispatch, including both intra-group and inter-group dependency checking. See column 26, lines 29-41. Note that the instructions held in DBUF 68 are grouped and dispatched for simultaneous execution if intra-group or inter-group hazards would not result from their simultaneous dispatch. Also, previously issued instruction groups will be checked against by "ready-to-issue" instructions (via inter-group dependency checks) until they propagate through the pipeline and complete (i.e. the result from the instruction is available and all dependent instructions can proceed). Therefore, the grouping and dependency checking occurs over plural execution cycles.

10. Referring to claim 10, Vegesna has taught a superscalar processor as described in claim 9. Vegesna has further taught:

a) grouping logic implementing plural early pipeline stages of the superscalar processor. From Fig.24 and column 22, lines 25-29, the basic concept of the system starts with IFETCH 4 fetching instructions. DBUF 68 is then used to hold and dispatch the fetched instructions. The instructions may or may not be grouped and dispatched simultaneously (dependency checks are required). See column 26, lines 29-41. It is further disclosed in column 4, lines 18-26, that the scheduling and issue functions are performed in the decode stage (D) of the pipeline. Therefore, grouping and issuing instructions would require two pipeline stages (plural early pipeline stages): The fetch stage (F) is the point when instructions are fetched and the decode stage (D) is where the fetched instructions are grouped (or not grouped depending on hazard detection) and issued.

b) plural execution units of varying pipeline depth coupled to receive instructions dispatched from the grouping logic. See Fig.13.

11. Referring to claim 11, Vegesna has taught a superscalar processor as described in claim 9. Vegesna has further taught that the instruction grouping identifies successive groups of instructions from an instruction stream for dispatch to respective ones of plural execution units of the superscalar processor. Recall from column 26, lines 29-41, that the scheduler will group and simultaneously issue the instructions in DBUF 68 if no inter/intra-dependencies exist. Furthermore, in column 22, lines 34-40, it is disclosed that a second buffer FBUF is used to replenish the DBUF with the next instructions so that the next instructions can be grouped and simultaneously issued, if possible. Therefore, the instruction grouping will continue to identify succeeding groups for as long as instruction fetching occurs.

12. Referring to claim 12, Vegesna has taught a superscalar processor as described in claim 11. Vegesna has further taught that the superscalar processor dispatches all instructions from a particular one of the successive groups before dispatching any instructions from a subsequent one of the successive groups. See column 22, lines 40-49, and note that execution is in-order and that instructions in a group prior to a subsequent group would be issued before instructions in the subsequent group.

13. Referring to claim 13, Vegesna has taught a superscalar processor as described in claim 9. Vegesna has further taught that the intra-group dependency checking spans at least two of the plural execution cycles. From column 26, lines 29-41, and column 22, lines 34-40, it should be noted that as long as instructions are fetched, instructions will be issued, and an issuing function of this system includes intra-group dependency checking. Therefore, this checking will span for as long as it is possible to simultaneously issue a group of instructions.

14. Referring to claim 14, Vegesna has taught a superscalar processor as described in claim 9. Vegesna has further taught that the intra-group dependency checking is independent of the inter-group dependency checking. See column 26, lines 29-43, and note that the dependency checks are done by separate circuitry so that they can be done in parallel.

15. Referring to claim 15, Vegesna has taught a superscalar processor as described in claim 9. Vegesna has further taught that the data dependency and resource allocation checks in earlier pipeline stages of instruction grouping are based, at least in part, on a predicted subsequent state of the superscalar processor. See column 2, lines 54-61, and note that a branch instruction (i.e. a program control instruction) can be issued simultaneously with another type of instruction. From Fig.10, it can be seen that when dealing with branches, a predict-not-taken scheme is

implemented so that the pipeline is not stalled. Due to this prediction, instructions from the mispredicted path may be fetched and issued, and until the true outcome of the branch is known, the processor will be in a predicted state, wherein resource allocation and dependency checks for the predicted target instructions will occur in the manner described in column 26, lines 29-41.

16. Referring to claim 16, Vegesna has taught a superscalar processor as described in claim 9. Vegesna has further taught that non-deterministic conditions are evaluated in a final stage of instruction grouping prior to dispatch. See column 26, lines 29-41. *The Free On-line Dictionary of Computing © 1993-2001* defines the term non-deterministic as "Exhibiting nondeterminism," where nondeterminism is defined by the same source as "A property of a computation which may have more than one result." One such condition that is checked would be intra-group dependencies. Assuming, no inter-group dependencies exist, if an intra-group dependency exists between the two instructions in DBUF 68, then only one of the instructions will be issued. On the other hand, if an intra-group dependency does not exist between the two instructions in DBUF 68, then both of the instructions will be issued simultaneously as a group. According to the definition above, such a dependency check would be associated with more than one result (i.e. simultaneous issuance or singular issuance).

17. Referring to claim 17, Vegesna has taught a processor comprising:

a) plural functional units that execute instructions in respective numbers of processor cycles. See Fig.13 and column 12, line 55, to column 13, line 14.

b) grouping logic coupled to the functional units and pipelined to compute, over plural cycles, T, of the processor, a future state, S(t + T), of the processor based on a prior state, S(t), of the processor and based thereon to select a group of instructions from a program sequence thereof

for dispatch to the functional units. See Fig.24 and column 26, lines 29-41, and recall that the grouping logic performs intra and inter-dependency checks. Through dependency computing (especially inter-dependency computing), the processor will check the current state S(t) of the system (i.e. dependencies between instructions ready to issue and instructions that have been issued but not completed), and through this check, the grouping logic will determine what instructions to issue in some future cycle wherein the state of the processor will be S(t + T). For example, if a floating-point addition instruction enters the floating-point addition execution unit at S(t) and the result will be available at S(t + 5), then the processor will know not to dispatch an instruction dependent on that floating-point addition instruction until S(t + 5).

18. Referring to claim 18, Vegesna has taught a processor as described in claim 17. Furthermore, the superscalar processor of claim 13 operates the same as the processor of claim 18. Therefore, claim 18 is rejected for the same reasons set forth in the rejection of claim 13.

19. Referring to claim 19, Vegesna has taught a processor as described in claim 17. Furthermore, the superscalar processor of claim 14 operates the same as the processor of claim 19. Therefore, claim 19 is rejected for the same reasons set forth in the rejection of claim 14.

20. Referring to claim 20, Vegesna has taught a processor as described in claim 17. Vegesna has further taught:

a) intra-group dependencies are checked by the pipelined grouping logic beginning in a first of the T cycles. See column 4, lines 18-26, column 26, lines 7-10, and column 26, lines 29-41, and note that intra-group dependency checking occurs in the Decode (D) stage, where the (D) stage is the first stage where dependency checking occurs.

b) non-deterministic dependency conditions are checked during a last of the T cycles. Again, *The Free On-line Dictionary of Computing © 1993-2001* defines the term non-deterministic as "Exhibiting nondeterminism," where nondeterminism is defined by the same source as "A property of a computation which may have more than one result." Such a condition would be exhibited by inter-group dependency checking, for example, since inter-group dependency checking will either result in non-dependent instructions being issued or dependent instructions being stalled (note that more than one result will occur based on the computation). And, from column 26, lines 35-41, it can be seen that inter-group dependency checks are performed between the "ready-to-issue" instructions in DBUF 68 and the instructions in every other stage of the pipeline after the decode (D) stage. This would include the Execute (E) stage, Cache (C) stage (more commonly known as the MEM stage), and the Write-back (W) stage. Therefore, such non-deterministic dependency checks will occur in the latter of the T stages (i.e. when instructions are in the (W) stage, for instance).

21. Referring to claim 21, Vegesna has taught a processor as described in claim 20. Vegesna has further taught that inter-group dependencies are checked independent of the intra-group dependencies. See column 26, lines 29-43, and note that the dependency checks are done by separate circuitry so that they can be done in parallel.

22. Referring to claim 22, Vegesna has taught a processor as described in claim 20. Vegesna has further taught that the grouping logic implements T stages of a pipeline of the processor. See column 26, lines 29-41, and note that instruction grouping and issuing is dependent on the inter-group dependency checks, which are performed between "ready-to-issue" instructions and instructions in every other stage of the pipeline after the (D) stage. Intra-group dependency

checks are actually performed in the (D) stage. Therefore, dependency checks are made in multiple stages of the pipeline (i.e. T stages).

23. Referring to claim 23, Vegesna has taught a processor as described in claim 20. Vegesna has further taught that T is four. According to column 26, lines 7-10 and 35-41, dependency checking occurs in the (D) stage and it involves instructions in all pipeline stages succeeding the (D) stage. From Fig.13, it can be seen that the dependency checking involves the (D) stage, (E) stage, (C) stage, and (W) stage. Therefore, T is equal to four stages.

24. Referring to claim 24, Vegesna has taught a processor as described in claim 17. Vegesna has further taught that the functional units include at least one functional unit capable of receiving and completing an instruction for each of the processor cycles. The ALU in Fig.6 performs basic integer arithmetic and logical operations that require at most one clock cycle to complete. Therefore, it can operate on a new instruction every processor cycle. See column 12, lines 57-59.

25. Referring to claim 25, Vegesna has taught a processor as described in claim 17. Vegesna has further taught that the functional units include at least one functional unit requiring multiple of the processor cycles for receiving and completing an instruction. Note the floating-point units described in column 12, line 66, to column 13, line 3.

26. Referring to claim 26, Vegesna has taught a method of operating a processor, the method comprising:

a) identifying successive groups of instructions for dispatch to respective ones of plural execution units of the processor. Recall from column 26, lines 29-41, that the scheduler will group and simultaneously issue the instructions in DBUF 68 to the execution units (for example

in Fig.13) if no inter/intra-dependencies exist. Furthermore, in column 22, lines 34-40, it is disclosed that a second buffer FBUF is used to replenish the DBUF with the next instructions so that the next instructions can be grouped and simultaneously issued, if possible. Therefore, the instruction grouping will continue to identify succeeding groups for as long as instruction fetching occurs.

b) performing, during plural pipelined execution cycles of the processor, dependency checking amongst instructions of a later one of the groups and between the instructions of the later group and instructions of a preceding one of the groups. Recall from column 26, lines 29-41 that intra-dependencies and inter-dependencies are checked, wherein intra-dependency checking involves checking amongst instructions of a particular group and inter-dependency checking involves checking between instructions of a later group and instructions of a preceding group.

c) dispatching instructions of the later group only after all instructions of the preceding group have been dispatched. See column 22, lines 40-49, and note that execution is in-order and that instructions in a group prior to a subsequent group would be issued before instructions in the subsequent group.

27. Referring to claim 27, Vegesna has taught a method of operating a processor as described in claim 26. Vegesna has further taught that the dependency checking is performed by pipelined grouping logic. From Fig.24 and column 22, lines 25-29, the basic concept of the system starts with IFETCH 4 fetching instructions. DBUF 68 is then used to hold and dispatch the fetched instructions. The instructions may or may not be grouped and dispatched simultaneously (dependency checks are required). See column 26, lines 29-41. It is further disclosed in column 4, lines 18-26, that the scheduling and issue functions are performed in the decode stage (D) of

the pipeline. Therefore, grouping and issuing instructions would require two pipeline stages: The fetch stage (F) is the point when instructions are fetched and the decode stage (D) is where the fetched instructions are grouped (or not grouped depending on hazard detection) and issued.

28. Referring to claim 28, Vegesna has taught a method of operating a processor as described in claim 26. Vegesna has further taught that intra-group dependency checking and within group resource allocation are performed in successive processor cycles by pipelined grouping logic. See column 26, lines 7-49. Note that dependency checking and resource allocation will occur for all processor cycles, including those following the current cycle. Therefore, these tasks would be performed in successive cycles (i.e. they're done in the current cycle and done in a successive cycle).

29. Referring to claim 29, Vegesna has taught a method of operating a processor as described in claim 26. Vegesna has further taught that intra-group dependency checking is performed independent of inter-group dependency checking. See column 26, lines 29-43, and note that the dependency checks are done by separate circuitry so that they can be done in parallel.

30. Referring to claim 30, Vegesna has taught a method of operating a processor as described in claim 26. Vegesna has further taught that non-deterministic conditions are evaluated in a final one of the pipelined execution cycles implemented by pipelined grouping logic. Again, *The Free On-line Dictionary of Computing © 1993-2001* defines the term non-deterministic as "Exhibiting nondeterminism," where nondeterminism is defined by the same source as "A property of a computation which may have more than one result." Such a condition would be exhibited by inter-group dependency checking, for example, since inter-group dependency checking will either result in non-dependent instructions being issued or dependent instructions being stalled

(note that more than one result will occur based on the computation). And, from column 26, lines 35-41, it can be seen that inter-group dependency checks are performed between the "ready-to-issue" instructions in DBUF 68 and the instructions in every other stage of the pipeline after the decode (D) stage. This would include the Execute (E) stage, Cache (C) stage (more commonly known as the MEM stage), and the Write-back (W) stage. Therefore, such non-deterministic dependency checks will occur in the latter of the T stages (i.e. when instructions are in the (W) stage, for instance).

31. Referring to claim 31, Vegesna has taught a method of grouping instructions for dispatch to execution units of a processor, the method comprising:

a) in a first cycle of processor execution, identifying plural candidate instructions for an instruction group. From Fig.24 and column 22, lines 25-29, the basic concept of the system starts with IFETCH 4 fetching instructions, which are eligible to be grouped. DBUF 68 is then used to hold and dispatch the fetched instructions.

b) in a subsequent cycle of processor execution, beginning intra-group dependency checking as amongst instructions of the instruction group. From column 26, lines 7-41, it is disclosed that in the (D) stage, subsequent to the (F) stage, intra-group dependencies are checked.

c) in a cycle of processor execution prior to dispatch of any instruction from the instruction group, checking non-deterministic conditions. See column 26, lines 29-41. Note that inter-group dependencies are checked in a cycle before they are dispatched.

d) in a cycle of processor execution prior to the non-deterministic dependency condition checking, initiating inter-group dependency checking as between instructions of the instruction group and instructions of one or more prior instruction groups. It is inherent that the inter-group

dependency checking must be initiated before it actually takes place. And, the fetch (F) stage in Vegesna's system would act as the cycle that actually initiates dependency checking since the fetch stage is what fills the DBUF 68. By filling the DBUF 68, the processor knows that non-deterministic checks must be performed. If instructions are no longer fetched in the fetch (F) stage, then the DBUF 68 will not be filled and non-deterministic checks will not be performed.

32. Referring to claim 32, Vegesna has taught a method as described in claim 31. Vegesna has further taught dispatching one or more instructions of the instruction group only after all instructions of the one or more prior instruction groups have been dispatched. See column 22, lines 40-49, and note that execution is in-order and that instructions in a group prior to a subsequent group would be issued before instructions in the subsequent group.

33. Referring to claim 33, Vegesna has taught a method as described in claim 31. Vegesna has further taught that the non-deterministic condition checking is performed conservatively, based on an assumption of no change in condition.

34. Referring to claim 34, Vegesna has taught a method as described in claim 31. Vegesna has further taught:

a) that non-deterministic condition checking is performed aggressively, computing dispatch conditions for at least two alternatives including no change in condition and condition resolution. It is the inherent function of inter-group dependency checks to provide two alternatives; the first is to allow dispatch of the dependent instruction when the dependency condition is resolved, and the second is to prevent issue of the dependent instruction when the dependency condition is unresolved.

b) a control signal is selective for a particular one of the computed alternatives.. See Fig.24 and column 28, lines 24-35. Note that signals 95 and 97 are used to specify whether or not the instructions can be dispatched.

35. Referring to claim 35, Vegesna has taught an apparatus comprising:

a) plural functional units. See Fig.13.

b) means for grouping, over plural pipeline stages, instructions for dispatch to respective ones of the functional units. See column 26, lines 29-41.

Referring to claim 36, Vegesna has taught an apparatus as described in claim 35. Furthermore, it has been noted that the processor of claim 17 performs the same function as the apparatus of claim 36. Therefore, claim 36 is rejected for the same reasons set forth in the rejection of claim 17.

Response to Arguments

36. Applicant's arguments filed on August 7, 2003, have been fully considered but they are not persuasive.

37. In the remarks, Applicant argues the rejection of claim 1 on pages 11-12 of the remarks, in substance that:

> "Vegesna simply does not perform its grouping and dependency checks over plural cycles. Indeed, quite to the contrary, Vegesna performs its analogous scheduling operations in a single pipeline stage (stage D) so that they will be completed within one CPU clock cycle."

38. These arguments are not found persuasive for the following reasons:

a) It has been noted by the examiner that instruction grouping and dependency checking does occur in the (D) stage of Vegesna. However, the applicant is reading the claim too narrowly.

More specifically, claim 1, as written, is not limited to the instruction grouping and dependency checking taking a plurality of execution cycles. Instead, an alternate interpretation (and the interpretation by the examiner) would be to read the claim as saying that instruction grouping and dependency checking occurs over multiple execution cycles. That is, instead of grouping and dependency checking taking multiple cycles to complete, the grouping and checking take one cycle but more grouping and checking will occur next cycle (i.e., over a plurality of cycles). For instance, by looking at Fig. 14(a) of Vegesna, one would see that there are 4 execution cycles (cycles 3-6). One of ordinary skill in the art would also recognize that if a particular application were made up of more than 8 instructions, then more than 4 (E) stages would be required. As a result, it should be realized that grouping and dependency checking would continue to occur for as long as there are instructions to schedule. And, this grouping and checking would occur over a plurality of execution cycles. For instance, looking at Fig. 14(a), assume that instructions iA and iB have issued in parallel (as shown). In cycle 3 (during the first execute cycle), instructions 1A+1 and 1B+1 will be checked for intra-dependencies and the instructions will be checked for inter-dependencies with the two previous instructions (iA and iB). If no dependencies exist, then the instructions can be issued together (as shown). Likewise, in cycle 4 (during the second execute cycle), instructions iA+2 and iB+2 will be checked for intra-dependencies and the instructions will be checked for inter-dependencies with the four previous instructions (iA, iB, iA+1, and iB+1). If no dependencies exist, then the instructions can be issued together (as shown). This would continue on and on over a plurality of execution cycles for as long as instructions need to be scheduled. Therefore, it can be seen that Vegesna does anticipate Applicant's claim 1 and all claims that include this form of claim language.

39. In the remarks, Applicant argues the rejection of claim 12, and other similar claims, on pages 12-13 of the remarks, in substance that:

> "In particular, the Office incorrectly states (quoting Applicant's claim language) that Vegesna 'dispatches all instructions from a particular one of the successive groups before dispatching any instructions from a subsequent one of the successive groups.' In point of fact, Vegesna, particularly col. 22, lines 40-49 thereof, is quite to the contrary. Indeed, that section of Vegesna describes that an unissued instruction in "SLOTB" of a given packet (DBUF) is shifted into the "SLOTA" position thereof and another valid instruction (i.e., a SLOTA instruction from the subsequent packet in FBUF) fills the SLOTB position of DBUF. In short, Vegesna merges an instruction from a subsequent packet into a preceding packet. SLOTA and SLOTB instructions can then be dispatched together."

40. These arguments are not found persuasive for the following reasons:

a) It has been noted by the examiner that the situation explained by the applicant above is a situation that occurs within the system of Vegesna. However, the applicant has failed to recognize other situations that occur within the system of Vegesna which read on the applicant's claims. For example, as disclosed in column 22, lines 40-49, of Vegesna, the DBUF has two slots (SLOTA and SLOTB) for holding two instructions which may or may not issue simultaneously (depending on dependencies). If only one of the instructions in DBUF can issue due to dependencies, then the SLOTA instruction is issued first. The SLOTB instruction will then be moved to SLOTA for possible issuing in the next cycle. SLOTB will then be refilled by an instruction that is waiting in the FBUF. In general, this is a display of in-order execution. If instructions could only issue one at a time due to dependencies, then the instruction in SLOTA would go, where the next oldest instruction will then be moved to SLOTA for issue in the next cycle (if possible). One situation that should be recognized by the applicant is that if an initial group of two instructions are in the DBUF together and they can be grouped for parallel dispatch, then these two instructions would be dispatched before any other instructions of a

subsequent group. On the other hand, if the SLOTA instruction is issued and the SLOTB instruction must wait, then the SLOTB instruction will be moved to SLOTA, while the next valid instruction from the FBUF will be moved to SLOTB. If the new instruction in SLOTB cannot issue in the next cycle with the old SLOTB instruction (which is now in SLOTA), then the SLOTA instruction will again issue singularly, resulting in all of the instructions from the initial group being issued before any instruction in the second group. Consequently, these situations allow Vegesna to read on the applicant's claims that involve this claim language..

Conclusion

41. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to David J. Huisman whose telephone number is (703) 305-7811. The examiner can normally be reached on Monday-Friday (8:00-4:30).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (703) 305-9712. The fax phone number for the organization where this application or proceeding is assigned is (703) 872-9306.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 305-3900.

DJH
David J. Huisman
August 13, 2003

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100